Da Teoria à Prática: Avaliando Soluções para o Problema do Caixeiro Viajante

Matheus Felipe Akira de Assis Oliveira

December 11, 2023

Abstract

Este artigo explora aspectos práticos de algoritmos para solucionar problemas desafiadores, focando na computação de rotas no Problema do Caixeiro Viajante (PCV). A avaliação abrange uma solução exata usando branch-and-bound e duas soluções aproximadas, twice-around-the-tree e o algoritmo de Christofides, ambos aplicáveis a instâncias euclidianas do PCV. O principal objetivo é entender os desafios inerentes à implementação de algoritmos ensinados em sala de aula.

1 Introduction

A otimização de rotas e a resolução eficiente do Problema do Caixeiro Viajante (PCV) têm sido desafios persistentes e essenciais em diversas áreas, desde logística até circuitos integrados. Nesse contexto, este trabalho busca não apenas explorar implementações práticas de algoritmos para solucionar o PCV, mas também proporcionar uma imersão na complexidade e nas decisões inerentes a essas implementações.

Ao avaliar algoritmos exatos, como o branch-and-bound, e aproximativos, como twice-around-the-tree e o algoritmo de Christofides, este trabalho visa analisar criticamente o desempenho de cada abordagem em termos de tempo, espaço e qualidade das soluções obtidas.

Portanto, este trabalho não apenas atende aos objetivos acadêmicos de aplicar conceitos teóricos na prática, mas também oferece uma visão crítica e informada sobre as implicações e desafios associados à implementação de algoritmos para o PCV em situações do mundo real.

2 Um pouco sobre o problema do caixeiro viajante

O Problema do Caixeiro Viajante (PCV) é um desafio clássico na área de otimização combinatória, amplamente estudado em ciência da computação e matemática. Ele se refere a uma situação em que um viajante deve visitar um conjunto de cidades, percorrendo a menor distância total possível e retornando ao ponto de origem. Apesar de sua formulação aparentemente simples, o PCV torna-se complexo à medida que o número de cidades aumenta, uma vez que o número de possíveis rotas cresce exponencialmente.

A complexidade do PCV reside na busca pela solução ótima entre todas as combinações de trajetos, uma vez que o número de possíveis itinerários aumenta fatorialmente com o número de cidades. Este problema é classificado como NP-difícil, o que significa que não há um algoritmo eficiente conhecido para resolvê-lo em tempo polinomial, tornando-o um dos problemas mais desafiadores em ciência da computação. No entanto, apesar de sua dificuldade, o PCV é de grande importância prática em logística, roteamento e outras áreas, incentivando a busca constante por algoritmos aproximados e heurísticas eficientes para encontrar soluções aceitáveis em tempo razoável.

Ao longo dos anos, o PCV tem sido abordado de diversas maneiras, com a aplicação de técnicas como algoritmos genéticos, algoritmos de otimização por enxame de partículas e métodos baseados em grafos. Essas abordagens buscam encontrar soluções próximas da ótima, mesmo que não garantam a obtenção da solução perfeita em tempo polinomial. A relevância contínua do PCV destaca-se na sua representação de desafios fundamentais em otimização, explorando as fronteiras da computação e matemática aplicada.

Figure 1: Exemplo de modelagem do PCV.

3 Datasets usado nesse trabalho

Neste estudo, utilizamos instâncias compiladas na biblioteca TSPLIB, uma referência reconhecida para problemas de otimização combinatória, especificamente no contexto do Problema do Caixeiro Viajante (PCV). Focamos em instâncias que empregam a função de custo baseada na distância euclidiana em 2D, proporcionando uma representação realista de problemas práticos.

O conjunto de teste consiste em instâncias selecionadas a partir do arquivo disponibilizado junto a esta descrição. Este arquivo fornece informações essenciais para cada instância, incluindo seu nome, número de nós (tamanho), e um limiar de qualidade, que representa a solução ótima segundo a biblioteca TSPLIB. Detalhes adicionais sobre as instâncias, como descrição do formato dos arquivos de entrada, podem ser encontrados no documento original em http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp95.pdf. O arquivo com a lista de instâncias utilizado neste trabalho está disponível em [https://github.com/MatheusAkira/Solutions-to-difficult-problems.git].

É relevante observar que o documento mencionado também contém informações sobre a descrição completa do formato dos arquivos de entrada utilizados para representar as instâncias do PCV. Estes arquivos, disponíveis em http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/, seguem padrões específicos que facilitam a leitura e processamento das instâncias pelos algoritmos implementados neste estudo.

A referência ao ótimo encontrado em http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/STSP.html destaca a qualidade da solução ideal para cada instância, fornecendo um critério valioso para avaliação de desempenho dos algoritmos aplicados. Ao utilizar este conjunto de teste da TSPLIB, buscamos não apenas desafiar os algoritmos propostos, mas também contribuir para a compreensão da eficácia destes em cenários realistas e benchmark reconhecido.

4 Algoritmos usados para resolver o PCV no trabalho

4.1 Branch and Bound

O algoritmo *Branch and Bound* (BB) é uma técnica de busca exata amplamente empregada para resolver problemas de otimização combinatória, incluindo o Problema do Caixeiro Viajante (PCV). Sua abordagem sistemática visa encontrar a solução ótima ao explorar de maneira eficiente o espaço de soluções viáveis.

No contexto do PCV, o BB inicia construindo uma árvore de busca, onde cada nó representa um subproblema. A cada nível da árvore, o algoritmo faz escolhas incrementais, representando diferentes ordens de visita às cidades. Ao explorar esses caminhos, o BB utiliza limites superiores e inferiores para podar ramos que não levariam a uma solução melhor do que a melhor já encontrada.

Uma característica notável do BB é sua garantia de otimalidade: ao final da execução, a solução encontrada é a ótima, uma vez que o algoritmo explora todas as possibilidades e elimina soluções subótimas durante o processo.

A complexidade do BB no PCV depende da estratégia de exploração da árvore de busca. A abordagem *depth-first* (busca em profundidade), adotada neste trabalho, oferece vantagens em termos de espaço, pois requer uma quantidade limitada de memória para armazenar os estados atuais da

busca. No entanto, a complexidade temporal do BB permanece exponencial no pior caso, dada a natureza combinatória do PCV.

A escolha da abordagem *depth-first* para o BB neste estudo foi motivada por sua eficiência na economia de espaço. Dada a grande quantidade de instâncias consideradas a partir da biblioteca TSPLIB, a limitação de espaço torna-se crucial. A estratégia *depth-first* permite explorar profundamente uma ramificação antes de passar para a próxima, minimizando a quantidade de memória necessária.

Apesar da complexidade temporal exponencial, o BB com abordagem *depth-first* é uma escolha viável para instâncias menores e médias do PCV, oferecendo uma combinação equilibrada entre garantia de otimalidade e eficiência espacial.

4.2 Twice-Around-the-Tree (2-aproximado)

O algoritmo *Twice-Around-the-Tree* (TAT) é uma heurística simples e eficaz para o Problema do Caixeiro Viajante (PCV). Inicialmente, o algoritmo constrói uma árvore geradora mínima (AGM) e, em seguida, duplica suas arestas. A solução é obtida ao percorrer a árvore resultante duas vezes, visitando cada vértice uma única vez. O TAT fornece soluções aproximadas e, embora não garanta otimalidade, é eficiente para instâncias com grande número de cidades.

O TAT possui complexidade temporal e espacial polinomial, tornando-o uma opção viável para instâncias de tamanho moderado. No entanto, a garantia de aproximação é limitada, sendo mais adequado para cenários onde soluções subótimas são aceitáveis.

Vantagens: - Simplicidade e facilidade de implementação. - Desempenho aceitável para instâncias grandes.

Desvantagens: - Não garante solução ótima. - Sensível à qualidade da AGM inicial.

4.3 Algoritmo de Christofides (1.5 aproximado)

O algoritmo de Christofides combina técnicas de aproximação para oferecer soluções eficientes e próximas à ótima para o PCV. Sua abordagem inclui a construção de uma solução inicial utilizando uma Árvore Geradora Mínima (AGM), a resolução do Problema do Emparelhamento Perfeito Mínimo e a geração de um circuito euleriano.

O algoritmo de Christofides possui complexidade temporal polinomial, tornando-o eficiente para instâncias moderadamente grandes. No entanto, sua complexidade espacial também é polinomial, garantindo um bom desempenho em termos de uso de memória.

Vantagens: - Garantia de aproximação de 3/2 da solução ótima. - Eficiência para instâncias de tamanho moderado.

Desvantagens: - Implementação mais complexa do que heurísticas simples. - Limitação na garantia de aproximação.

A taxa de aproximação é uma medida que compara a solução produzida pelo algoritmo com a solução ótima conhecida. Para o TAT, a taxa de aproximação pode variar dependendo da qualidade da AGM inicial, enquanto o algoritmo de Christofides oferece uma garantia de aproximação de 3/2, indicando que a solução encontrada será, no máximo, 50 por cento pior do que a ótima.

5 Metodologia

5.1 Coleta de Dados

- Baixar os datasets indicados para o estudo do Problema do Caixeiro Viajante (PCV).
- Certificar-se de obter datasets relevantes e representativos para a análise.

5.2 Exploração de Dados

- Carregar os datasets e examinar suas características principais.

5.3 Ordenação por Tamanho

- Ordenar os datasets com base na quantidade de nós (vértices) em cada instância do PCV.

5.4 Visualização de Dados

- Criar visualizações gráficas para representar a distribuição dos tamanhos dos datasets.
- Utilizar gráficos, histogramas ou outros métodos visuais para apresentar as características dos datasets ordenados.

5.5 Análise Comparativa

- Comparar as características dos datasets ordenados.

5.6 Preparação para Experimentos

- Definir claramente as métricas de desempenho que serão utilizadas para avaliar os algoritmos de solução do PCV.

5.7 Implementação dos Algoritmos

- Implementar os três algoritmos para o Problema do Caixeiro Viajante: *Branch and Bound*, *Twice-Around-the-Tree*, e o algoritmo de Christofides.
- Certificar-se de que a implementação seja consistente e bem documentada.

5.8 Avaliação de Desempenho

- Aplicar os algoritmos aos datasets ordenados por tamanho.
- Registrar e analisar os resultados em termos de métricas de espaço, consumo e qualidade do resultado.

5.9 Análise de Resultados

- Analisar e interpretar os resultados obtidos.
- Comparar o desempenho dos algoritmos em diferentes tamanhos de problemas do PCV.

6 Resultados

Vamos analisar a seguir os resultados pelas seguintes perspectivas. Tempo de execução, memmória usada e qualidade da solucação.

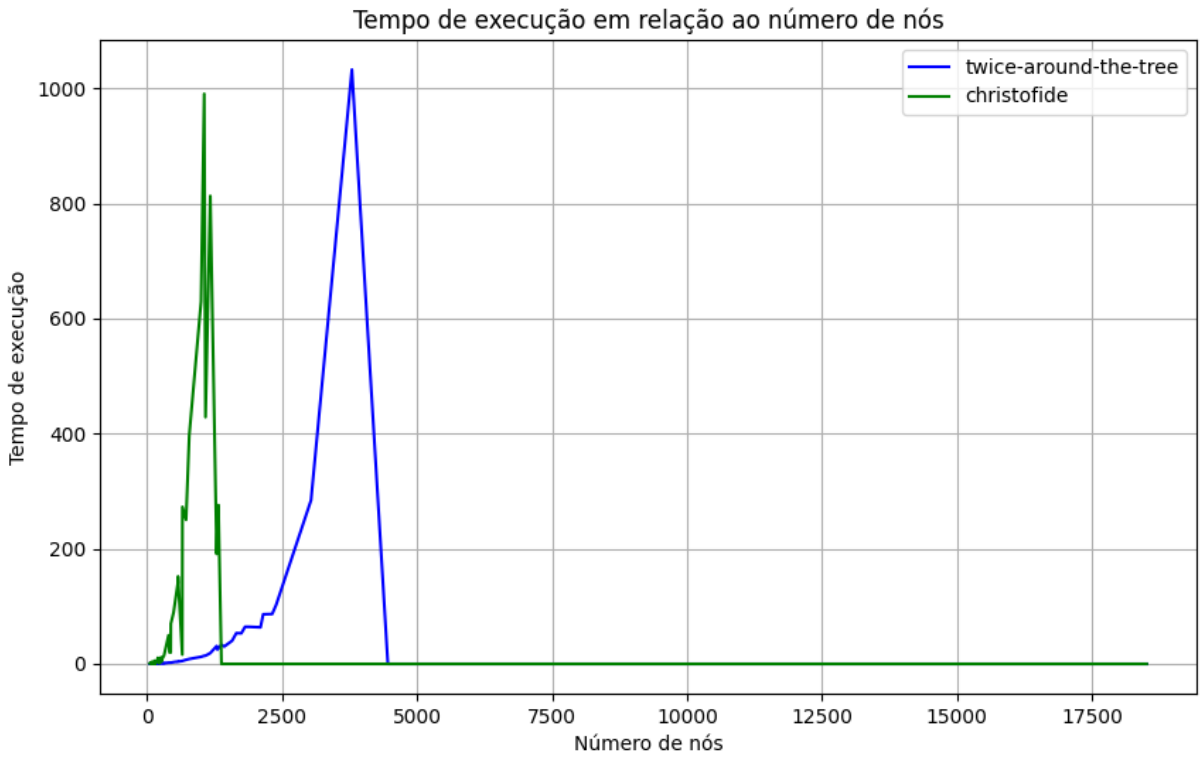

Figure 2:

Em relação ao tempo de execução, o Branch and bound não executou nenhuma instancia em menos de 30min. Então ele não foi representado no gráfico. Dos algoritmos que rodaram dentro do tempo estabelecido, podemos perceber que o twice around the tree executa bem mais rápido que o Christofides. Ademais, ele consegue calcular mais instâncias do que o Christofides dentro do tempo estabelecido. Os pontos em que o tempo é 0 significam que o algoritmo não rodou para aquela quantidade de nó.

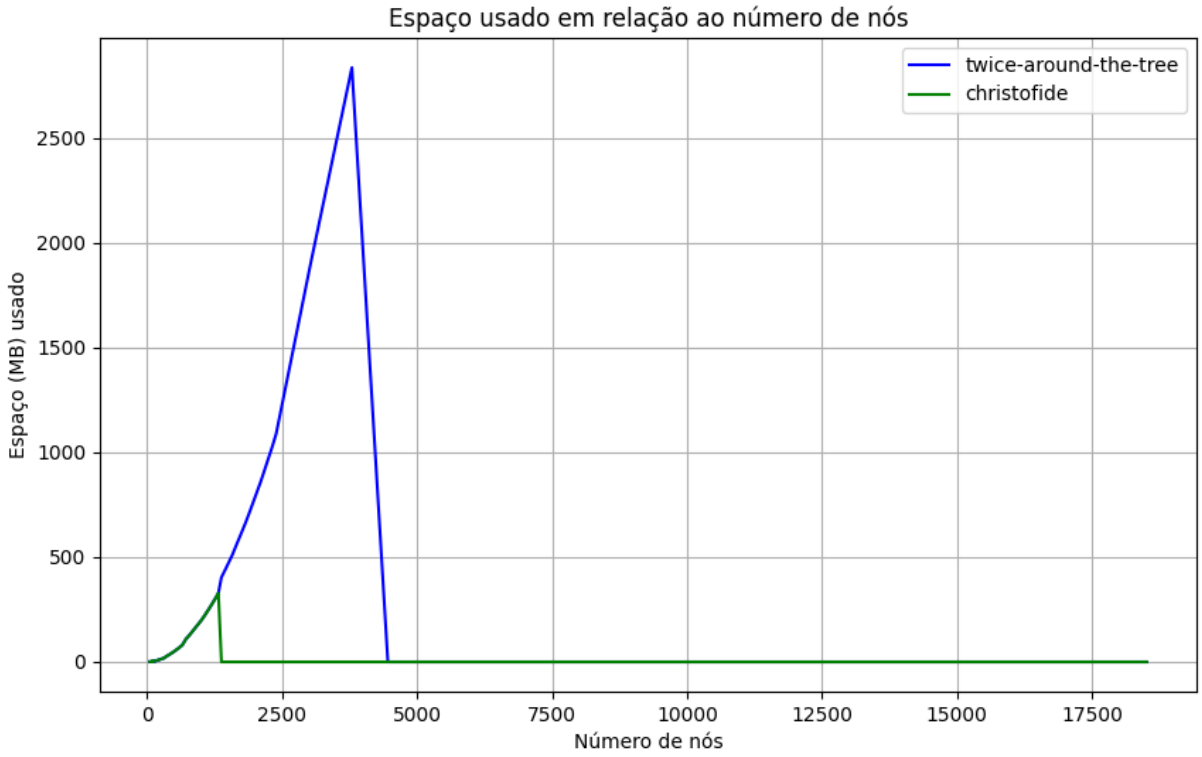

Figure 3:

Podemos perceber na (Figura 3) que o Christofides e o twice around the tree possuem o mesmo consumo de espaço em relação a quantidade de nós que recebem como entrada. Como o branch-and-bound não rodou nos experimentos, ele não foi representado.

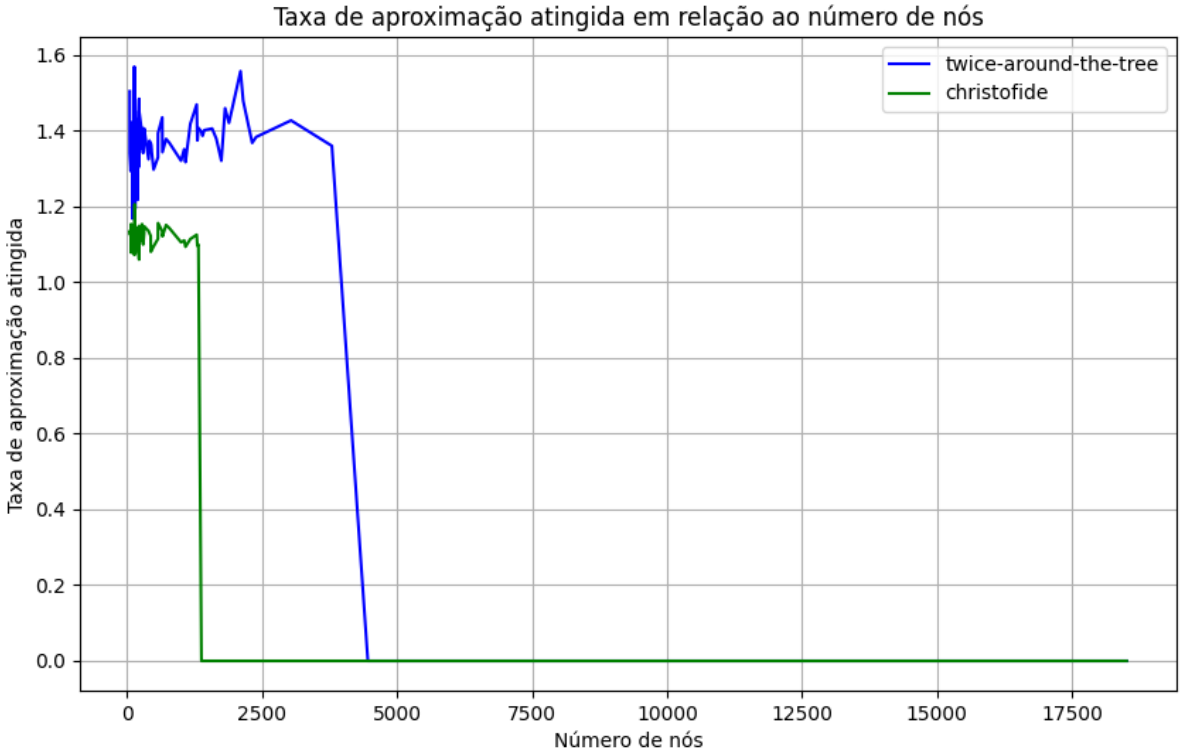

Figure 4:

Na (Figura 4) podemos ver que o Christofides tem uma qualidade de solução melhor em relação ao twice around the tree. O branch-and-bound, ficaria em uma taxa de aproximação 1, ou seja, solução ótima, caso tivesse rodado nos experimentos.

7 Conclusões

O branch-and-bound possui solução ótima, porém podemos observar que é um algoritmo que demanda muito e não foi possível rodar as instâncias dos datasets com esse algoritmo. Em relação a qualidade da solução ele é o melhor, mas demora muito tempo para ser executado.

O Christofides demora menos que o branch-and-bound, mas o resultado da qualidade da solução é inferior, ou seja, ele é 1.5 aproximado. Isso signica que dos três algoritmos, ele fica em segundo lugar em relação a qualidade. Em relação ao twice around the tree ele demora mais para executar em uma mesma instância.

Por fim, temos o twice around the tree. Ele é o mais rápido dos três algoritmos propostos. Porém, é o que tem a pior qualidade de solução. Em relação ao espaço ocupado durante a execução desse algoritmo. Ele empata com o Christofides de acordo com o experimentos feitos,

.

References

[1] Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G., & Shmoys, D. B. *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. John Wiley & Sons, 1985.

[2] Applegate, D. L., Bixby, R. E., Chvátal, V., & Cook, W. J. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, 2006.

[3] Little, J. D. C., Murty, K. G., Sweeney, D. W., & Karel, C. *An algorithm for the traveling salesman problem.* Operations Research, 11(6), 972-989, 1963.

[4] Lawler, E. L. *A Procedure for Computing the K Best Solutions to Discrete Optimization Problems and Its Application to the Shortest Path Problem.* Management Science, 12(4), 355-374, 1966.

[5] Christofides, N. *Worst-case analysis of a new heuristic for the traveling salesman problem.* Technical Report 388, Graduate School of Industrial Administration, Carnegie Mellon University, 1976.